callHORIZONS Documentation

Release 1.0.5

Michael Mommert

Jul 19, 2018

Contents

CALLHORIZONS is a Python interface to access JPL HORIZONS ephemerides and orbital elements of Solar System bodies.

Please note that CALLHORIZONS is not maintained anymore. Please use astroquery.jplhorizons instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.

CHAPTER 1

Installation

Please note that CALLHORIZONS is not maintained anymore. Please use [astroquery.jplhorizons](http://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html) **instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.**

1.1 PIP

Using PIP, simply call:

```
pip install callhorizons
```

1.2 GitHub

Using git, simply clone the code from GitHub:

```
git clone https://github.com/mommermi/callhorizons
```

or, download and unpack the zip file with all the files from GitHub and then run from the callhorizons directory:

```
python setup.py install
```

CHAPTER 2

How to Use It?

Please note that CALLHORIZONS is not maintained anymore. Please use [astroquery.jplhorizons](http://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html) **instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.**

1. import the module into your code:

```
import callhorizons
```

2. initialize a QUERY object with an objectname that is readable by the JPL HORIZONS website; this might be the target asteroid's name:

```
dq = callhorizons.query('Don Quixote')
```

number:

```
dq = callhorizons.query('3552')
```

name and number:

```
dq = callhorizons.query('(3552) Don Quixote')
```

designation:

```
dq = callhorizons.query('1983 SA')
```

or packed designation:

```
dq = callhorizons.query('J83S00A')
```

Comet names may be the full periodic number and name:

```
dq = callhorizons.query('1P/Halley')
dq = callhorizons.query('3D/Biela')
```

periodic number only:

```
dq = callhorizons.query('9P')
```

orbit solution ID:

```
dq = callhorizons.query('900190')
```

temporary designation:

```
dq = callhorizons.query('P/2001 YX127')
```

temporary designation with name:

```
dq = callhorizons.query('P/1994 N2 (McNaught-Hartley)')
```

or long period / hyperbolic comet designation, with or without name:

```
dq = callhorizons.query('C/2013 US10 (Catalina)')
dq = callhorizons.query('C/2012 S1')
```

Fragments may also be requested:

```
dq = callhorizons.query('C/2001 A2-A')
dq = callhorizons.query('73P-C/Schwassmann Wachmann 3 C')
```

but note that the name is ignored. The following will not return fragment B, but instead the ephemeris for 73P (compare with the previous example):

```
dq = callhorizons.query('73P/Schwassmann Wachmann 3 B')
```

By default, comet queries will return the most recent or current apparition (HORIZONS's 'CAP' parameter). This behavior can be disabled with the *cap=False* keyword argument:

```
dq = callhorizons.query('9P', cap=False)
```

If there are multiple orbit solutions available, CALLHORIZONS will raise a `ValueError` and provide the URL with explanations.

You can also query **major bodies** (planets and moons) and **spacecraft**. This is a little bit trickier, since there are no clear naming conventions for these objects, causing ambiguities (see the Horizons documentation for a discussion). Assume that we want to select Jupiter's moon Io, we would could use the following line:

```
io = callhorizons.query('Io', smallbody=False)
```

Please note the flag `smallbody=False`, which is necessary here. However, this line will cause an error when one tries to obtain ephemerides or orbital elements: `ValueError: Ambiguous target name; check URL: http://ssd.jpl.nasa.gov/....` Calling the provided URL explains the problem. Horizons selects all known objects that contain the letters *io*. In order to unambiguously refer to Jupiter's moon Io, one has to use the provided ID number, which is 501. Hence, one should use:

```
io = callhorizons.query(501, smallbody=False)
```

Every target listed in Horizons provides an ID number, allowing for unambiguous identification. If there is ambiguity - or if the target is not in the Horizons database - CALLHORIZONS will raise a `ValueError` and provide the URL with explanations. Spacecraft can be selected the same way, also requiring the `smallbody=False` flag.

3. set the time range of epochs that you want to query using:

```
dq.set_epochrange('2016-02-27 03:20', '2016-02-28 05:20', '1h')
```

where the order is *start date and time*, *end date and time*, and *step size* using *YYYY-MM-DD HH:MM* UT times, or set discrete times:

```
dq.set_discreteepochs([2457446.177083, 2457446.182343])
```

where discrete epochs are provided in the form of a list of Julian Dates.

4. query ephemerides for the given times for a given observatory code (here: 568, Mauna Kea):

```
dq.get_ephemerides(568)
```

or, obtain the target's orbital elements:

```
dq.get_elements()
```

The queried data are stored in the *QUERY* object and can be accessed easily:

```
dq.fields    # provide list of available target properties
dq['RA']     # access 'RA' for all epochs
dq[0]        # access all properties for the first epoch
dq.dates     # provide list of epochs
dq.query     # show URL to query Horizons
```

Queried data can also be filtered, e.g., based on airmass:

```
dq[dq['airmass'] < 1.5]
```

Orbital elements queried with CALLHORIZONS can be directly converted into PyEphem objects to calculate the ephemerides:

```
import ephem
dq.get_elements()
dq_pyephem = dq.export2pyephem()
```

Once ephemerides or orbital elements have been queried, the URL with which HOrizons has been called can be listed:

```
print(dq.query)
```

This is especially useful for debugging and finding out why a query might have failed.

For more information, see the *Examples* and the *callhorizons* reference.

CHAPTER 3

Examples

Please note that CALLHORIZONS is not maintained anymore. Please use [astroquery.jplhorizons](http://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html) **instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.**

1. Find the hours on the night of 2015-10-25 (UT) when Centaur Echeclus is observable with airmass < 1.5 from Mauna Kea (observatory code: 568) during dark time:

```
import callhorizons
echeclus = callhorizons.query('echeclus')
echeclus.set_epochrange('2015-10-25', '2015-10-26', '1h')
echeclus.get_ephemerides(568)
print(echeclus[(echeclus['solar_presence'] != 'daylight') & (echeclus['airmass']
↪< 1.5)]['datetime'])
```

   Note: you can also use HORIZONS' own `skip daylight` function and set an airmass limit during the query:

```
echeclus.get_ephemerides(568, skip_daylight=True, airmass_lessthan=1.5)
print(echeclus['datetime'])
```

2. Pull the orbital elements of Saturn on a specific date:

```
import callhorizons
saturn = callhorizons.query('Saturn', smallbody=False)
saturn.set_discreteepochs('2451234.5')
saturn.get_elements()
```

   This will cause a `ValueError: Ambiguous target name; check URL: ...`. Why did that happen? Check out the URL that is provided with the error message; it will tell you the reason. The target name is ambiguous, since there is (the center of) Saturn and the barycenter of the Saturn system. If you are interested in the planet, use the ID number (699) instead of the planets name:

```
import callhorizons
saturn = callhorizons.query('699', smallbody=False)
```

(continues on next page)

(continued from previous page)

```
saturn.set_discreteepochs('2451234.5')
saturn.get_elements()
```

3. more examples will come in the future . . . (what are you interested in?)

CHAPTER 4

callhorizons

Please note that CALLHORIZONS is not maintained anymore. Please use [astroquery.jplhorizons](http://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html) **instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.**

4.1 callhorizons module

Please note that CALLHORIZONS is not maintained anymore. Please use [astroquery.jplhorizons](http://astroquery.readthedocs.io/en/latest/jplhorizons/jplhorizons.html) **instead, which will be maintained in the future and offers additional functionality. I apologize for any inconvenience.**

CALLHORIZONS - a Python interface to access JPL HORIZONS ephemerides and orbital elements.

This module provides a convenient python interface to the JPL HORIZONS system by directly accessing and parsing the HORIZONS website. Ephemerides can be obtained through get_ephemerides, orbital elements through get_elements. Function export2pyephem provides an interface to the PyEphem module.

michael.mommert (at) nau.edu, latest version: v1.0.5, 2017-05-05. This code is inspired by code created by Alex Hagen.

- v1.
- v1.0.5: 15-epoch limit for set_discreteepochs removed
- v1.0.4: improved asteroid and comet name parsing
- v1.0.3: ObsEclLon and ObsEclLat added to get_ephemerides
- v1.0.2: Python 3.5 compatibility implemented
- v1.0.1: get_ephemerides fixed
- v1.0: bugfixes completed, planets/satellites accessible, too
- v0.9: first release

class callhorizons.**query**(*targetname*, *smallbody=True*, *cap=True*, *nofrag=False*, *comet=False*, *asteroid=False*)

Bases: object

__dict__ = mappingproxy({'fields': <property object>, '__dict__': <attribute '__dict__

__getitem__(*key*)
provides access to query data

Parameters **key** – str/int; epoch index or property key

Returns query data according to key

__init__(*targetname*, *smallbody=True*, *cap=True*, *nofrag=False*, *comet=False*, *asteroid=False*)
Initialize query to Horizons

Parameters

- **targetname** – HORIZONS-readable target number, name, or designation
- **smallbody** – boolean use smallbody=False if targetname is a planet or spacecraft (optional, default: *True*); also use *True* if the targetname is exact and should be queried as is
- **cap** – set to *True* to return the current apparition for comet targets
- **nofrag** – set to *True* to disable HORIZONS's comet fragment search
- **comet** – set to *True* if this is a comet (will override automatic targetname parsing)
- **asteroid** – set to *True* if this is an asteroid (will override automatic targetname parsing)

Returns None

__len__()
returns total number of epochs that have been queried

__module__ = 'callhorizons'

__repr__()
returns brief query information

__str__()
returns information on the current query as string

__weakref__
list of weak references to the object (if defined)

dates
returns list of epochs that have been queried (format 'YYYY-MM-DD HH-MM-SS')

dates_jd
returns list of epochs that have been queried (Julian Dates)

export2pyephem(*center='500@10'*, *equinox=2000.0*)
Call JPL HORIZONS website to obtain orbital elements based on the provided targetname, epochs, and center code and create a PyEphem (http://rhodesmill.org/pyephem/) object. This function requires PyEphem to be installed.

Parameters

- **center** – str; center body (default: 500@10 = Sun)
- **equinox** – float; equinox (default: 2000.0)

Result list; list of PyEphem objects, one per epoch

Example

```
>>> import callhorizons
>>> import numpy
>>> import ephem
>>>
>>> ceres = callhorizons.query('Ceres')
>>> ceres.set_epochrange('2016-02-23 00:00', '2016-02-24 00:00', '1h
↪')
>>> ceres_pyephem = ceres.export2pyephem()
>>>
>>> nau = ephem.Observer() # setup observer site
>>> nau.lon = -111.653152/180.*numpy.pi
>>> nau.lat = 35.184108/180.*numpy.pi
>>> nau.elevation = 2100 # m
>>> nau.date = '2015/10/5 01:23' # UT
>>> print ('next rising: %s' % nau.next_rising(ceres_pyephem[0]))
>>> print ('next transit: %s' % nau.next_transit(ceres_pyephem[0]))
>>> print ('next setting: %s' % nau.next_setting(ceres_pyephem[0]))
```

fields
returns list of available properties for all epochs

get_elements(*center='500@10'*, *asteroid=False*, *comet=False*)
Call JPL HORIZONS website to obtain orbital elements based on the provided targetname, epochs, and center code. For valid center codes, please refer to http://ssd.jpl.nasa.gov/horizons.cgi

Parameters **center** – str; center body (default: 500@10 = Sun)

Result int; number of epochs queried

Example

```
>>> ceres = callhorizons.query('Ceres')
>>> ceres.set_epochrange('2016-02-23 00:00', '2016-02-24 00:00', '1h
↪')
>>> print (ceres.get_elements(), 'epochs queried')
```

The queried properties and their definitions are:

Property	Definition
targetname	official number, name, designation [string]
H	absolute magnitude in V band (float, mag)
G	photometric slope parameter (float)
datetime_jd	epoch Julian Date (float)
e	eccentricity (float)
p	periapsis distance (float, au)
a	semi-major axis (float, au)
incl	inclination (float, deg)
node	longitude of Asc. Node (float, deg)
argper	argument of the perifocus (float, deg)
Tp	time of periapsis (float, Julian Date)
meananomaly	mean anomaly (float, deg)
trueanomaly	true anomaly (float, deg)
period	orbital period (float, Earth yr)
Q	apoapsis distance (float, au)

get_ephemerides(*observatory_code*, *airmass_lessthan=99*, *solar_elongation=(0, 180)*, *skip_daylight=False*)

Call JPL HORIZONS website to obtain ephemerides based on the provided targetname, epochs, and observatory_code. For a list of valid observatory codes, refer to http://minorplanetcenter.net/iau/lists/ObsCodesF.html

Parameters

- **observatory_code** – str/int; observer's location code according to Minor Planet Center
- **airmass_lessthan** – float; maximum airmass (optional, default: 99)
- **solar_elongation** – tuple; permissible solar elongation range (optional, deg)
- **skip_daylight** – boolean; crop daylight epoch during query (optional)

Result int; number of epochs queried

Example

```
>>> ceres = callhorizons.query('Ceres')
>>> ceres.set_epochrange('2016-02-23 00:00', '2016-02-24 00:00', '1h
↪')
>>> print (ceres.get_ephemerides(568), 'epochs queried')
```

The queried properties and their definitions are:

Property	Definition
targetname	official number, name, designation [string]
H	absolute magnitude in V band (float, mag)
G	photometric slope parameter (float)
datetime	epoch date and time (str, YYYY-MM-DD HH:MM:SS)
datetime_jd	epoch Julian Date (float)
solar_presence	information on Sun's presence (str)
lunar_presence	information on Moon's presence (str)
RA	target RA (float, J2000.0)
DEC	target DEC (float, J2000.0)
RA_rate	target rate RA (float, arcsec/s)
DEC_rate	target RA (float, arcsec/s, includes cos(DEC))
AZ	Azimuth meas East(90) of North(0) (float, deg)
EL	Elevation (float, deg)
airmass	target optical airmass (float)
magextinct	V-mag extinction due airmass (float, mag)
V	V magnitude (comets: total mag) (float, mag)
illumination	fraction of illuminated disk (float)
EclLon	heliocentr. ecl. long. (float, deg, J2000.0)
EclLat	heliocentr. ecl. lat. (float, deg, J2000.0)
ObsEclLon	obscentr. ecl. long. (float, deg, J2000.0)
ObsEclLat	obscentr. ecl. lat. (float, deg, J2000.0)
r	heliocentric distance (float, au)
r_rate	heliocentric radial rate (float, km/s)
delta	distance from the observer (float, au)
delta_rate	obs-centric radial rate (float, km/s)
lighttime	one-way light time (float, s)

Continued on next page

Table 1 – continued from previous page

Property	Definition
elong	solar elongation (float, deg)
elongFlag	app. position relative to Sun (str)
alpha	solar phase angle (float, deg)
sunTargetPA	PA of Sun->target vector (float, deg, EoN)
velocityPA	PA of velocity vector (float, deg, EoN)
GlxLon	galactic longitude (float, deg)
GlxLat	galactic latitude (float, deg)
RA_3sigma	3sigma pos. unc. in RA (float, arcsec)
DEC_3sigma	3sigma pos. unc. in DEC (float, arcsec)

isasteroid()
True if *targetname* appears to be an asteroid.

iscomet()
True if *targetname* appears to be a comet.

isorbit_record()
True if *targetname* appears to be a comet orbit record number.

NAIF record numbers are 6 digits, begin with a '9' and can change at any time.

parse_asteroid()
Parse *targetname* as if it were a asteroid.

Returns (string or None, int or None, string or None); The designation, number, and name of the asteroid as derived from *self.targetname* are extracted into a tuple; each element that does not exist is set to *None*. Parenthesis in *self.targetname* will be ignored. Packed designations and numbers are unpacked.

Example the following table shows the result of the parsing:

targetname	(desig, number, name)
1	(None, 1, None)
2 Pallas	(None, 2, Pallas)
(2001) Einstein	(None, 2001, Einstein)
1714 Sy	(None, 1714, Sy)
2014 MU69	(2014 MU69, None, None)
228195. 6675 P-L	(6675 P-L, 228195, None)
4101 T-3	(4101 T-3, None, None)
4015 Wilson-Harrington (1979 VA)	(1979 VA, 4015, Wilson-Harrington
J95X00A	(1995 XA, None, None)
K07Tf8A	(2007 TA418, None, None)
G3693	(None, 163693, None)
2017 U1	(None, None, None)

parse_comet()
Parse *targetname* as if it were a comet.

Returns (string or None, int or None, string or None); The designation, number and prefix, and name of the comet as derived from *self.targetname* are extracted into a tuple; each element that does not exist is set to *None*. Parenthesis in *self.targetname* will be ignored.

Example the following table shows the result of the parsing:

targetname	(desig, prefixnumber, name)
1P/Halley	(None, '1P', 'Halley')
3D/Biela	(None, '3D', 'Biela')
9P/Tempel 1	(None, '9P', 'Tempel 1')
73P/Schwassmann Wachmann 3 C	(None, '73P', 'Schwassmann Wachmann 3 C')
73P-C/Schwassmann Wachmann 3 C	(None, '73P-C', 'Schwassmann Wachmann 3 C')
73P-BB	(None, '73P-BB', None)
322P	(None, '322P', None)
X/1106 C1	('1166 C1', 'X', None)
P/1994 N2 (McNaught-Hartley)	('1994 N2', 'P', 'McNaught-Hartley')
P/2001 YX127 (LINEAR)	('2001 YX127', 'P', 'LINEAR')
C/-146 P1	('-146 P1', 'C', None)
C/2001 A2-A (LINEAR)	('2001 A2-A', 'C', 'LINEAR')
C/2013 US10	('2013 US10', 'C', None)
C/2015 V2 (Johnson)	('2015 V2', 'C', 'Johnson')
C/2016 KA (Catalina)	('2016 KA', 'C', 'Catalina')

query
returns URL that has been used in calling HORIZONS

set_discreteepochs (*discreteepochs*)
Set a list of discrete epochs, epochs have to be given as Julian Dates

Parameters **discreteepochs** – array_like list or 1D array of floats or strings

Returns None

Example

```
>>> import callhorizons
>>> ceres = callhorizons.query('Ceres')
>>> ceres.set_discreteepochs([2457446.177083, 2457446.182343])
```

set_epochrange (*start_epoch*, *stop_epoch*, *step_size*)
Set a range of epochs, all times are UT

Parameters

- **start_epoch** – str; start epoch of the format 'YYYY-MM-DD [HH-MM-SS]'
- **stop_epoch** – str; final epoch of the format 'YYYY-MM-DD [HH-MM-SS]'
- **step_size** – str; epoch step size, e.g., '1d' for 1 day, '10m' for 10 minutes...

Returns None

Example

```
>>> import callhorizons
>>> ceres = callhorizons.query('Ceres')
>>> ceres.set_epochrange('2016-02-26', '2016-10-25', '1d')
```

Note that dates are mandatory; if no time is given, midnight is assumed.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

Symbols

C

D

E

F

G

I

P

Q

S